

# Directory Services

Julia Ertl

The bioinformatics lab SS 2012

15.05.2012

# The Problem: grown structures



- redundancy information
- difficult accessing and sharing users or applications
- how to simultaneously access data from various sources

**Example:** Login at Cip Pool Amalienstraße

→ where to store user login information - at the network, at each server?

# The Solution: Naming and Directory Service

## **Naming Service:**

= give names to objects, objects can lie on any machine accessible from your network

→ mapping of human-recognizable identifier to a system-internal identification or address



## **Directory Service:**

like above but provides additional information → using attributes



- defines the namespace for the network
- hold one or more objects as named entries
- easily extendeable, store variety
- **directory design:**
  - have a set of rules
  - determines how network resources are named and identified
  - names are unique and unambiguous
  - specialized database with a hierarchical schema

# Directory Service - Relational Database ?

<b>Directory Service</b>	<b>Relational Database</b>
Optimized for read and search	Write-intensive operations
Static data	Data in flux or historical data
Object-oriented	ACID transactions
Hierarchical	Complex data models.
Multi-master replication	Data integrity

# Simple Directory Services

## System databases

- name service database like '/etc/passwd', '/etc/shadow', '/etc/group', '/etc/hosts'
- unix command **getent**: retrieves entries from system databases
- different configuration databases configured in **nsswitch.conf** → Name Service Switch

- **‘/etc/passwd’** : stores essential information, which is required during login i.e. user account information

Username: Password: UID: GID: Info: Home: Shell

ertlj:x:1000:1000:Julia Ertl,,,:/home/ertlj:/bin/bash

- **‘/etc/shadow’** : stores actual password in encrypted format

Username: Password : D0C : MinD : MaxD: Warn:Exp:Dis:Res

ertlj:\$6\$yQKwFbWJ\$0w7Q1:15329:0:99999:7:::

- **'/etc/passwd'** : stores essential information, which is required during login i.e. user account information

Username: Password: UID: GID: Info: Home: Shell

ertlj:x:1000:1000:Julia Ertl,,,:/home/ertlj:/bin/bash

- **'/etc/shadow'** : stores actual password in encrypted format

Username: Password : DOC : MinD : MaxD: Warn:Exp:Dis:Res

ertlj:\$6\$yQKwFbWJ\$0w7Q1:15329:0:99999:7:::

DOC - Day of last change of password, since 1.1.1970; MinD/MaxD - Min./Max. days password is valid; Warn - days until end of password lifetime

- **'/etc/group'** : stores group information or defines the user groups

Groupname: Password: GID: GroupList

bpw09\_9:x:19089:ansorge,ertlj,biosuper,biohyper

- **'/etc/hosts'**: list of known hosts (in the local network)

IP-address hostdomain [aliasname]

127.0.0.1 localhost

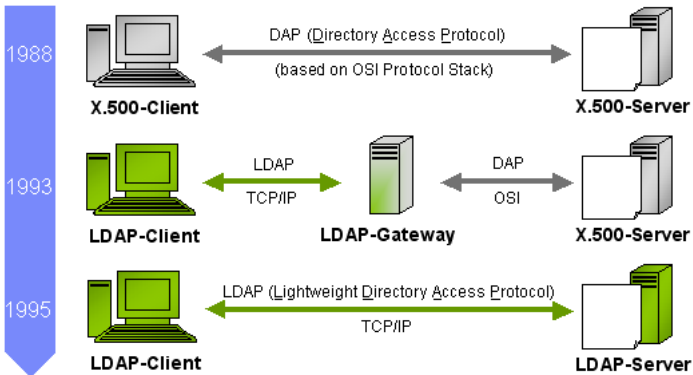
127.0.1.1 Linux-Home-Julia

# Directory Services: Server softwares

- **DNS - Domain Name System:** computer hostnames into IP addresses
- **NIS - Network Information Service:** client-server directory service protocol for user information storage
- historical: **NetInfo** for NEXT computers
  
- **LDAP/X.500** based implementations
  - OpenLDAP
  - Active Directory (Windows)
  - Apple Open Directory
  - OpenDS (java implementation)



- = Lightweight Directory Access Protocol
- = client-server protocol for accessing and managing directory information



# LDAP: Directory structure

- Schemas, objectClasses, Attributes and entries
- entry consists of a set of attributes
- each entry has a: **Unique Distinguished Name (DN)**
- first attribute(s) defines **Relative Distinguished Name (RDN)** component of the **DN** for each entry

Example:

`/foo/bar/myfile.txt` = DN and `myfile.txt` = RDN

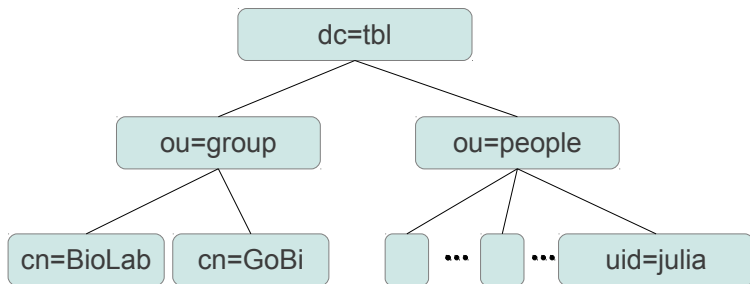
# LDAP: common attributes

<b>Name</b>	<b>Alias</b>	<b>objectClass</b>
cn	commonName	person organizationalPerson ...
sn	surname	person
dc	domainComponent	dcObject
o	organizationName	organization
ou	organisationalUnitName	organizationalUnit
uid	userid	account inetOrgPerson posixAccount

# LDAP: common objectClasses

Name	Must-Have attributes
account	userid
posixAccount	cn \$ uid \$ uidNumber gidNumber \$ homeDirectory
dcObject	dc
person	sn \$ cn
inetOrgPerson	-
organizationalPerson	-
organizationalUnit	ou

# Do a directory information tree (DIT)

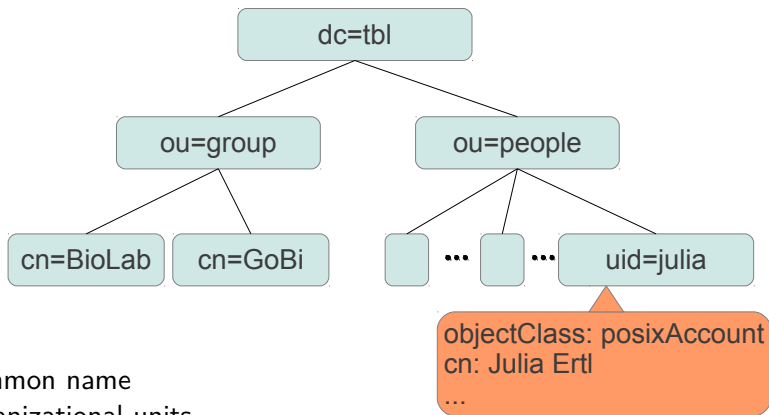


cn: common name

ou: organizational units

dc: domain component

# Do a directory information tree (DIT)



cn: common name

ou: organizational units

dc: domain component

# LDIF: LDAP Data Interchange Format

= transform directory structure in readable textfile → domain-name.ldif

```
dn: dc=tbl
objectClass: top
objectClass: dcObject
objectClass: organizationalRole
dc: tbl
description: tbl domain for us
```

```
dn: uid=julia,ou=people,dc=tbl
objectClass: posixAccount
cn: Julia Ertl
uid: julia
uidNumber: 1013
gidNumber: ...
...
```

# Configure the LDAP server daemon 'slapd' at runtime

- OpenLDAP and slapd = LDAP daemon (listens for LDAP connections)
- dynamic changes are saved when slapd is running from a slapd.d configuration directory
- solution: choose appropriate database type → cn=config.

# How do you control access to your LDAP database?

```
access to <what> [ by <who> <accesslevel> <control> ]+
```

```
olcAccess: {0}to * by dn.exact="gidNumber=0+uidNumber=0,  
cn=peercred,cn=external,cn=auth" manage  
by * break
```

# How do you control access to your LDAP database?

```
access to <what> [ by <who> <accesslevel> <control> ]+
```

```
olcAccess: {0}to * by dn.exact="gidNumber=0+uidNumber=0,  
cn=peercred,cn=external,cn=auth" manage  
by * break
```

# How do you control access to your LDAP database?

```
access to <what> [ by <who> <accesslevel> <control> ]+
```

```
olcAccess: {0}to * by dn.exact="gidNumber=0+uidNumber=0,  
cn=peercred,cn=external,cn=auth" manage  
by * break
```

# How do you control access to your LDAP database?

```
access to <what> [ by <who> <accesslevel> <control> ]+
```

```
olcAccess: {0}to * by dn.exact="gidNumber=0+uidNumber=0,  
cn=peercred,cn=external,cn=auth" manage  
by * break
```

# How do you control access to your LDAP database?

```
access to <what> [ by <who> <accesslevel> <control> ]+
```

```
olcAccess: {1}to attrs=userPassword,shadowLastChange  
            by self write  
            by anonymous auth  
            by dn="cn=admin,dc=tbl" write  
            by * none
```

# How do you control access to your LDAP database?

```
access to <what> [ by <who> <accesslevel> <control> ]+
```

```
olcAccess: {2}to dn.base="" by * read
```

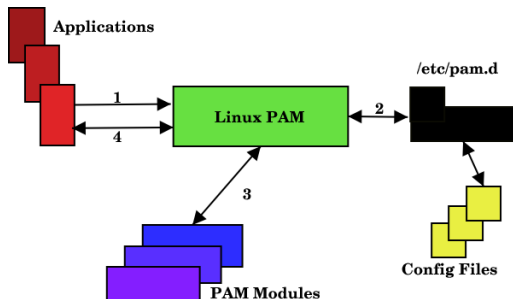
# How do you control access to your LDAP database?

```
access to <what> [ by <who> <accesslevel> <control> ]+
```

```
olcAccess: {3}to *      by self write  
                      by dn="cn=admin,dc=tbl" write  
                      by * read
```

# Pluggable authentication modules (PAM)

- allows user authentication → stored in a modular authentication library
- purely passive and must be always called by an application



# Pluggable authentication modules (PAM) and LDAP

## **pam\_ldap module:**

provides for authentication, authorization and password changing against LDAP servers

do: 'pam-auth-update'

Good working with

